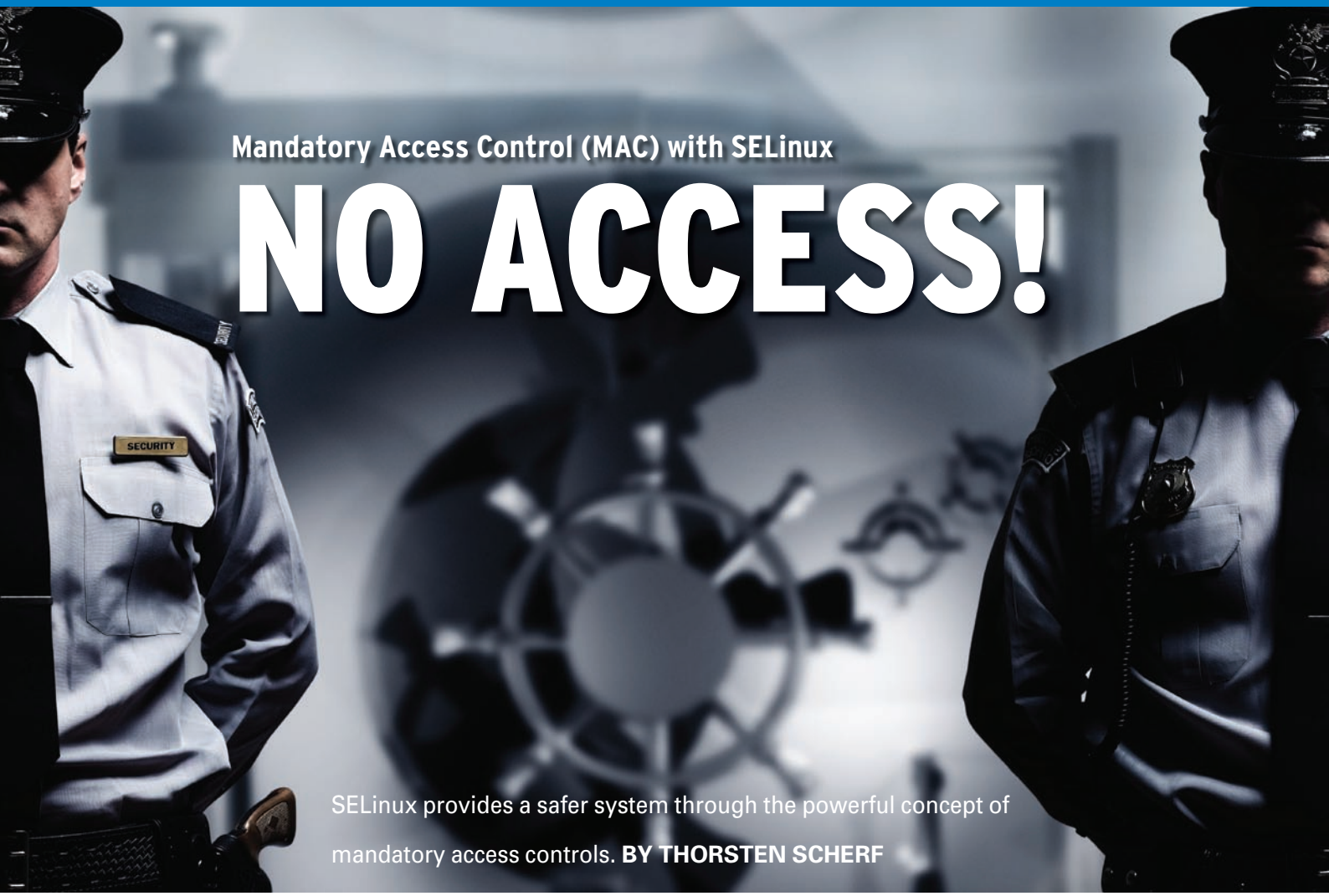

Mandatory Access Control (MAC) with SELinux

NO ACCESS!

SELinux provides a safer system through the powerful concept of mandatory access controls. **BY THORSTEN SCHERF**

Linux is an extremely safe operating system, but legacy access privileges provide no protection against misconfiguration or badly programmed software. If a program runs haywire because the administrator has forgotten to install the latest patch, or if a user escalates privileges due to an incorrect setting, the native safety of the system is no protection. SELinux mitigates the potential danger by adding an extra level of access control called the Mandatory Access Control (MAC) level.

About seven years ago, the National Security Agency (NSA) [1] launched the first version of SELinux. Intended as an extension for kernel 2.4 at the time, the kernel patches have since found their way into the official 2.6 kernel. For many distributions, SELinux is part of the standard configuration. The examples introduced in this article are based on the Red Hat community distribution Fedora Core 8, although they are generically valid on any other platform that supports SELinux. The important thing is that the required kernel support (*CONFIG_SECURITY_SELINUX*) and the *libselinux*, *policycoreutils*, and *selinux-policy-targeted* packages are installed. SELinux also requires a few standard packages (*SysV-Init*, *pam*, *util-linux*, *coreutils*, and others).

The legacy Linux security system is based on Discretionary Access Controls (DACs). This means that the owner of a file has absolute control over the object they have created. If a user inadvertently grants global write access to the file, a separate process that validates this step does not exist.

Or, if an attacker manages to execute arbitrary program code on the web server by exploiting a vulnerability in the web server software, the program code – a shell for example – will run with the privileges of a user account the server is running on. If that account is the *apache* user, the attacker has access to all files that the *apache* user account can access.

In most cases, no process checks whether the web server really needs to access the files that it is accessing to be able to do its chores. An attacker who gains access to the system then may be able to escalate their privileges on the machine. Just recently, many Linux systems were compromised because

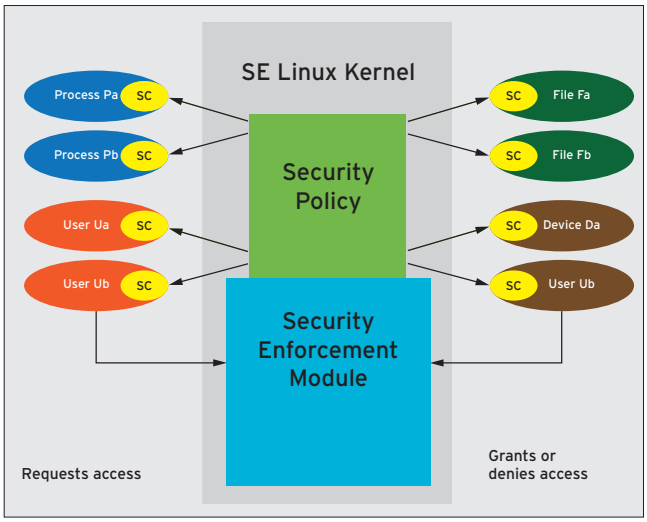

Figure 1: The kernel-based security server decides on the level of access.

of a bug in the kernel that allowed attackers to exploit the *vmsplice()* system call. As a consequence, attackers gained complete control over the whole system.

On systems with SELinux, every file, or every object, is given a separate security label by MAC for an extra level of control. In the case of file objects, Linux stores this label in the extended attributes. At the same time, each process, or each subject, is also given the corresponding label. This label, which is known as the security context, typically comprises three components: *User:Role:Type/Domain*. Two examples of this for the Apache server process file */usr/bin/httpd* file are as follows:

```
# ls -lZ /usr/sbin/http
-rwxr-xr-x  root ➥
 root system_u:object_r:➥
httpd_exec_t   /usr/sbin/httpd
```

and:

```
# ps -AZ|grep httpd
user_u:system_r:httpd_t  2571 ? ➥
       00:00:01 httpd
```

Just like security labels, the Posix ACLs extend attributes of a file. In this example, they look like this:

```
# getfattr -d -m ➥
security /usr/sbin/httpd
...
security.selinux=➥
"system_u:object_r:httpd_➥
exec_t\000"
```

SE Policy

The policy is another important component. An SE policy defines access between individual objects and subjects. The policy specifies which objects the process (such as an httpd process with a specific role) is allowed to access. If access is not explicitly permitted, it is initially logged and finally prohibited – at least in enforcing mode.

The kernel's security server makes sure that no infringement against the policy occurs. The security server is an entity that references the policy, and the security label defines whether access is permitted. To avoid performance overheads, the kernel-based server uses the Access Vector Cache (AVC).

File System Security Label

Although SELinux will work on file systems that do not support extended attributes, such as NFS or ISO9660, administrators have to resort to various workarounds to get this to happen. The *mount* command has an option to handle this: *context=<security-label>*. This lets you specify a security label for the complete filesystem.

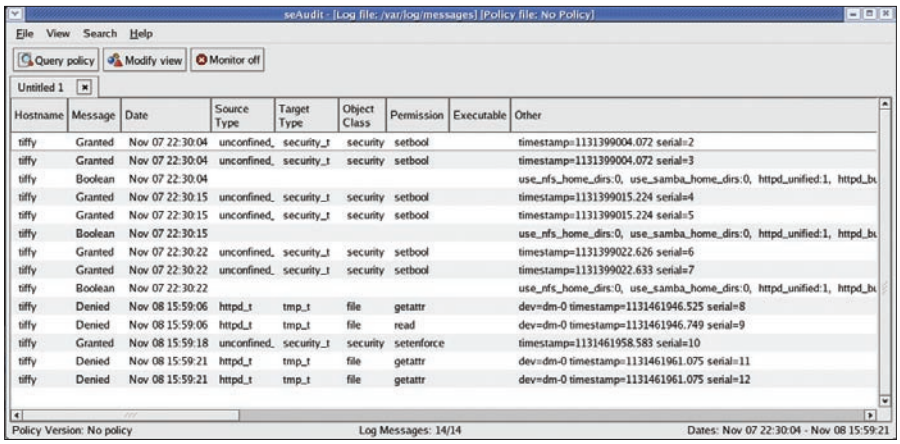

Figure 2: The sedaudit tool displays all the log entries for SELinux in a sorted list.

Demo

As a short demonstration of the way type enforcement works, consider the following: The administrator creates a file called *index.html* in the */tmp* folder. After doing so, the administrator moves (doesn't copy) the file into the web server's document root, which, on Fedora, is */var/www/html/*. Finally, the administrator launches the web server (*/etc/init.d/httpd start*) and accesses the page just

SELinux Functions

SELinux distinguishes three different implementations:

- Type Enforcement (TE)
- Role-Based Access Control (RBAC)
- Multi-Level Security (MLS)

TE specifies which subject is permitted to access which objects – for example, which process is allowed to access which files. However, a wide variety of different objects exists, including network ports or memory areas. SELinux assigns a domain to each subject and a type to each object. To explain this generically, type enforcement controls which domain is allowed to access which types. Both types and domains are identified in a similar way; they always end with *_t* (e.g., *httpd_t*).

RBAC uses an abstract user model and assigns each user exactly one role. Users then inherit the privileges assigned to the role. It is thus possible to assign a role to the root user that does not possess any administrative privileges. To change to another role with extended privileges, the user would first need to authenticate with a password. This is an interesting feature if you want to set up the machine without the omnipotent root user. The user can only assume a single role at any given time; however, users can give the *newrole* command (which is similar to *su*) to change roles, assuming the policy allows this. A typical role name is *user_r* (roles always end with *_r*).

Russell Coker offers a couple of SELinux play machines on the network [2] to allow people to experiment; these machines demonstrate RBAC functionality. Coker has published the root account for these machines, allowing anybody who is interested to log in as the administrator. Testers will soon find out that the command set available to them is extremely restricted, in that the *root* user is not the administrative user on these play machines.

Finally, MLS defines different security levels and is primarily used in high-security environments, such as military applications. Objects are assigned to security levels (confidential, strictly confidential, secret, and so on), and subjects are given permissions for these different levels of confidentiality. Wherever MLS is deployed, the security label is extended by adding a fourth and fifth component until it looks like this:

User:Role:Type/
Domain:Sensitivity:Category

created in a web browser (*http://localhost/index.html*). If SELinux enforcing mode is enabled (default), the web browser displays an error message because *index.html* was created in */tmp* and inherited its security label:

```
# ls -lZ /tmp/index.html
-rw-r--r--  root     root  ➥
root:object_r:tmp_t ➥
  /tmp/index.html
```

In contrast, the web server process runs in the domain titled *httpd_t*:

```
# ps -AZ|grep httpd
user_u:system_r:httpd_t➥
2571 ?➥
00:00:02  httpd
```

The policy needs a rule that gives the *httpd_t* domain access to *tmp_t* type files, but it does not exist. The web server needs to access its own configuration files, CGI scripts, and other content in its directory. Specific types exist for all of these files; for example, *httpd_config_t*, *httpd_log_t*, *httpd_sys_script_exec_t*, and *httpd_sys_content_t*.

Files in the */tmp* folder are not typically the kinds of objects the web server accesses; thus, no allow instruction is in the policy file for the *tmp_t* type.

If you check the */var/log/audit/audit.log* file, you will find a message to the effect that an unauthorized attempt to open a file has just taken place:

```
... audit(1202241301.521:12): ➥
avc: denied
{getattr } for  pid=6608 ➥
comm="httpd" name=➥
"index.html" dev=dm-0 ino=179881
scontext=user_u:system_r:➥
httpd_t tcontext=➥
root:object_r:tmp_t tclass=file
```

This log entry reveals that the process with PID 6608 and name *httpd* has just attempted to issue the *getattr* system call (i.e., it attempted to call the attributes) for a file with inode number 179881 and name *index.html*. *scontext* and *tcontext* refer to the security label for the source process (*apache*) and the target file (*index.html*). *avc: denied* indicates that the security server running in the kernel prevented this action.

If you prefer a more structured approach, you can use *seaudit* to view logfiles (Figure 2); the tool displays individual log entries graphically.

By calling *seaudit-report --html logfile*, you can even generate an HTML page that displays both the logs and various statistics on the SELinux system (see Figure 3).

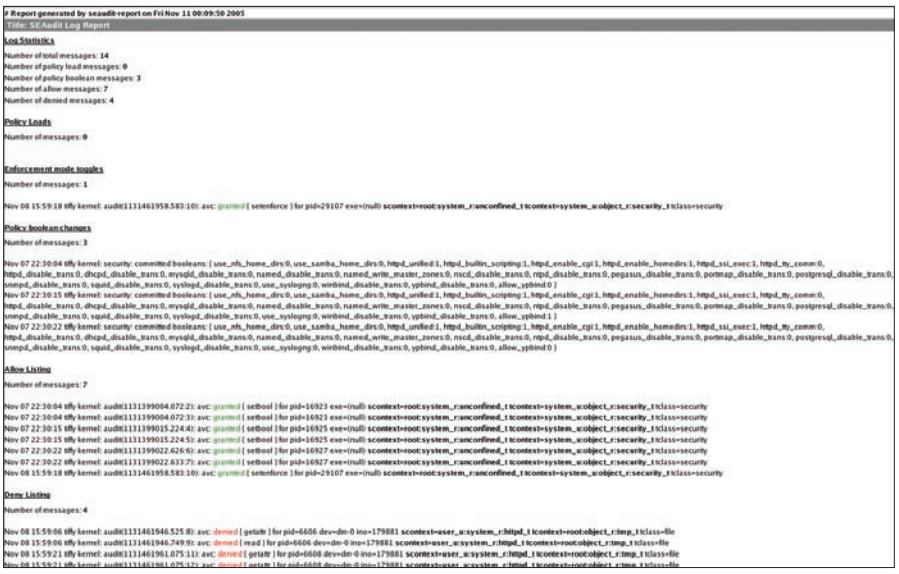

Figure 3: seaudit-report can generate statistics in HTML.

Spotting Trouble

If *setroubleshootd* is running, the following additional entry is displayed in the logfile */var/log/messages*:

```
setroubleshoot: #012 ➥
   SELinux is ➥
preventing the /usr/sbin/httpd ➥
from using potentially ➥
mislabeled files ➥
(/var/www/html/index.html).➥
#012 For complete ➥
SELinux messages➥
run sealert -l 5e982b3b-3ae7-➥
4848-b8bd-7d8553a07732
```

The *setroubleshoot* demon was developed some time back to make the slightly cryptic messages issued by the audit daemon more readable and to give users tips on resolving problems. If the user issues the command specified in the log entry, they get to see the explanation shown in Listing 1.

The Gnome desktop displays a small icon (a yellow shield) in the taskbar whenever an SELinux log entry is created. Users can click the icon to pop up the graphical SELinux troubleshooting browser, which lets you browse graphically through the processed messages

Listing 1: sealert -l

```
Summary
    SELinux is preventing the /usr/sbin/httpd from using potentially
    mislabeled files (/var/www/html/index.html).
Detailed Description
    SELinux has denied /usr/sbin/httpd access to potentially mislabeled
    file(s) (/var/www/html/index.html). This means that SELinux will
    not allow /usr/sbin/httpd to use these files. It is common for users
    to edit files in their home directory or tmp directories and then
    move (mv) them to system directories.  The problem is that the files
    end up with the wrong file context which confined applications are
    not allowed to access.
Allowing Access
    If you want /usr/sbin/httpd to access this files, you need to relabel
    them using restorecon -v /var/www/html/index.html.  You might want to
    relabel the entire directory using restorecon -R -v /var/www/html.
...
```

Customised IT solutions for individual needs are our strength!

With over 500 installed clusters, we offer everything – except inflexibility. We cover the entire portfolio from cost-effective entry clusters to enterprise solutions.

transtec Entry HPC Clusters are suitable for small work groups and fulfill the need for superior computing performance.
Our mid-range clusters ensure that there are sufficient resources available for multiple power users or mid-sized work groups.
The Enterprise Clusters offer the computing performance of multiple TFLOPs and a large number of compute nodes.
You will always have enough power for future requirements and to work efficiently.

>> www.transtec.co.uk/go/cluster

transtec

Any questions? We are happy to help:

• transtec Computers Ltd. • Suite A, Castle Link • 39 North Bar • Banbury • Oxfordshire OX16OTH
• Tel: +44 (0) 1295/756 100 • www.transtec.co.uk

Microsoft CERTIFIED Partner

Novell

intel Xeon inside Powerful. Efficient.

AMD Smarter Choice

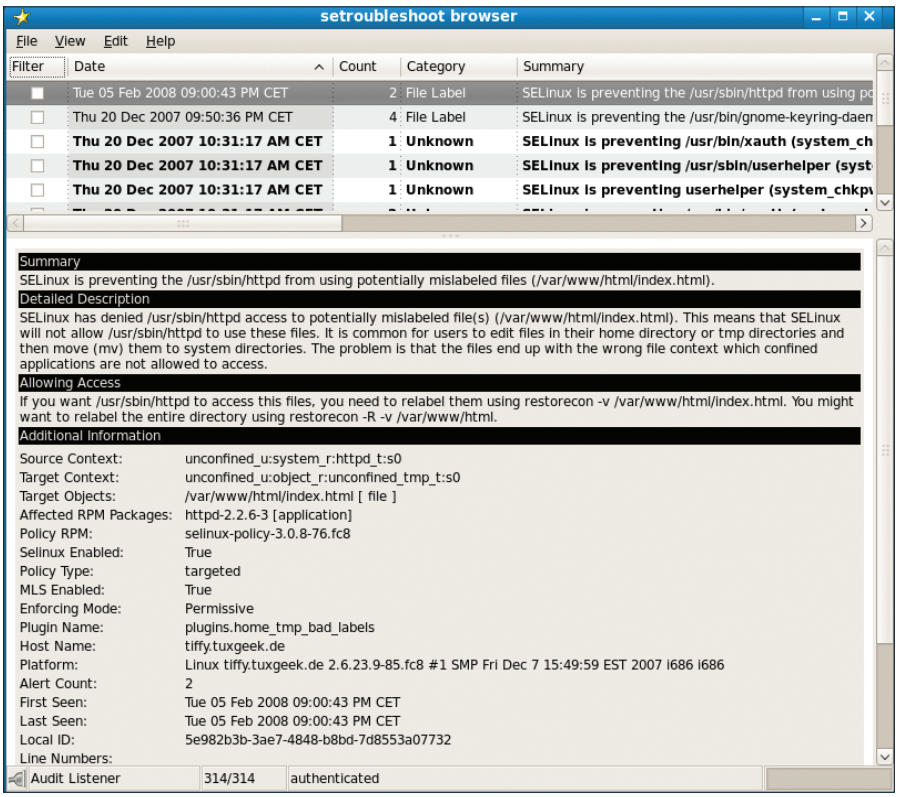

Figure 4: setroubleshootd offers a graphical front end for logs.

(Figure 4). This tool provides novice admins a way to resolve problems:

```
restorecon -v ➥
/var/www/html/index.html
```

The command uses a policy to set the correct label for the *index.html* file. Alternatively, the administrator can specify the correct file type manually:

```
chcon -t ➥
httpd_sys_content_t ➥
/var/www/html/index.html
```

Either way, the results should be:

```
# ls -lZ /var/www/➥
html/index.html
-rw-r--r--  root ➥
root system_u:object_r:➥
httpd_sys_content_t  ➥
/var/www/html/index.html
```

The next time somebody tries to display a file in the web browser, there shouldn't be any security obstacles.

The example here shows how SELinux works. Independent of legacy privileges, Linux only permits access if a corresponding entry in the SELinux policy exists (MAC). The security-conscious distributor or administrator will only create this entry if access is really necessary.

Administration

Various tools are available for managing an SELinux system. For example, a utility called *getenforce* displays the current SELinux mode. *setenforce 0|1* lets users change the mode, where *0* represents permissive mode and *1* stands for enforcing mode. Permissive mode means that unauthorized actions are logged but not prohibited, which is useful if you are developing a new policy module. The security server references its policy entries to decide what is permitted. To change a mode permanently, you need an entry in the */etc/selinux/config* file (Listing 2).

The most interesting tool for SELinux has to be *system-config-selinux* (Figure 5). It lets admins perform basic settings, such as the SELinux mode, while supporting more complex tasks such as creating new policy modules. You can also configure Booleans, which are simply instructions that enable policy rules you have prepared, without needing the *m4* macro language to do so (the whole policy is based on this language).

A variety of predefined Booleans exist; for example, you can allow a web server to access data in user folders (*UserDir*), or allow the name server to perform changes to the zone file (*DDNS*). Booleans can typically be displayed at the command line with *getsebool*. The *getsebool -a | grep httpd* command, for example, lists all the Booleans for the Apache web server (Listing 3).

A number of man pages describe the Booleans for the most popular network services. The *httpd_selinux* page helps you with the web server.

Also, you can change Booleans at the command line with *setsebool*. The following command line allows the web server to execute CGI scripts:

```
setsebool -P httpd_enable_cgi 1
```

sestatus is an interesting command-line tool that summarizes the current SELinux configuration (Listing 4).

Whereas RHEL 4 had a purely monolithic policy, today, modular variants of the policy are used. This gives administrators a number of advantages. For example, a policy developer no longer needs to worry about the complete policy sources for the SELinux system; it is sufficient to develop a single module for the application you want to protect and to add this module to the system.

Listing 2: /etc/selinux/config

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#       targeted - Targeted processes are protected,
#       mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

The default policy, which is part of the Fedora distribution (Targeted Policy), protects many applications out of the box. The programs protected by the policy are referred to as targeted programs, which explains the policy name. The *semodule* command returns all available policy modules (Listing 5).

If administrators want to remove a module, and thus SELinux protection for this program, they can simply pass in the module's name with the *-r* switch:

```
semodule -r amavis
```

Doing so permanently removes protection for the specified application, but you can reintroduce the module later. The Policy RPM stores all available standard modules in the */usr/share/selinux/targeted* directory. An administrator can reload the Amavis module by calling *semodule* as follows:

```
semodule -i ➥
/usr/share/selinux/targeted/➥
amavis.pp
```

The command reloads the Amavis module on the security server running on the kernel. The module's rule set then assumes responsibility for denying or permitting any actions that relate to the Amavis software. If you need a precise overview of which rules the individual modules include, you can install the SRPM (Source RPM) for the policy you are using, or you can simply install the graphical *apol* tool, which can display binary policy files in clear text format,

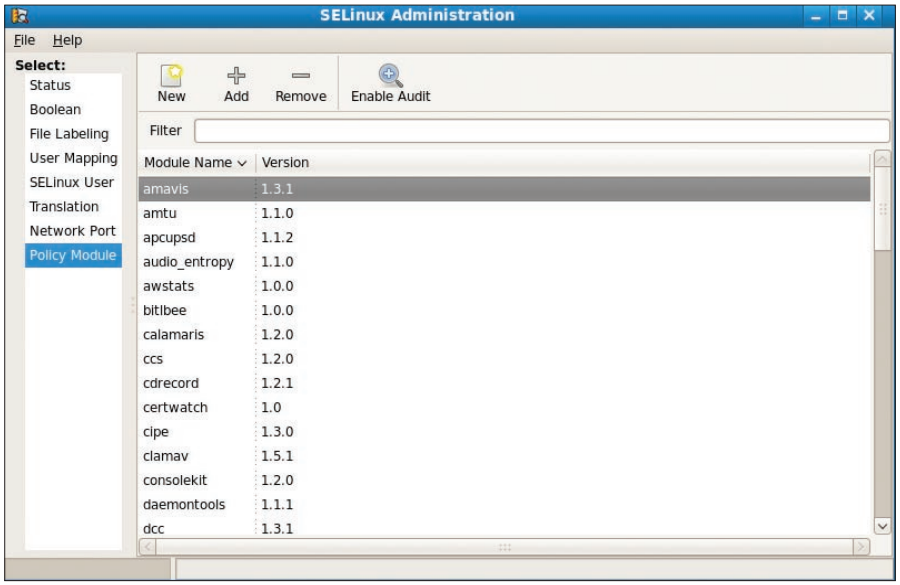

Figure 5: The system-config-selinux graphical configuration tool guarantees convenient administration of an SELinux system.

thus letting you investigate the deployed policy.

If this sounds too complicated, or if you simply need a generic overview of the policy that you have deployed, *seinfo* is your tool of choice (Listing 6). You can easily see how complex the total rule set actually is.

One feature of the SELinux policy on Fedora 8 is that it now also contains the properties of the older strict policy. To be more precise, it is now possible to use the targeted policy to restrict user accounts (i.e., to implement RBAC). For example, Dan Walsh has released a policy module titled *xguest* [3]. The module lets the administrator quickly convert any Gnome desktop into a kiosk system. The user is allowed to login as *xguest*. This user has very limited privileges on the Gnome desktop and a very restricted selection of programs. For example, network access is restricted to the Firefox web browser; all other applications do not have access to the network.

The module also prohibits modifications to the Gnome settings (*gconf*). This is an ideal environment for kiosk systems, such as those typically found at airports and in hotel lobbies. The *xguest* policy module can also serve as a starting point for your own development. For example, you could add further instructions to the existing rules to support SSH-based access.

Listing 3: getsebool -a | grep httpd

```
allow_httpd_anon_write --> off
allow_httpd_dbus_avahi --> off
allow_httpd_mod_auth_pam --> off
allow_httpd_sys_script_anon_
write --> off
httpd_builtin_scripting --> on
httpd_can_network_connect -->
off
httpd_can_network_connect_db -->
off
httpd_can_network_relay --> off
httpd_can_sendmail --> off
httpd_enable_cgi --> on
httpd_enable_ftp_server --> off
httpd_enable_homedirs --> on
httpd_ssi_exec --> off
httpd_tty_comm --> on
httpd_unified --> on
httpd_use_cifs --> off
httpd_use_nfs --> off
```

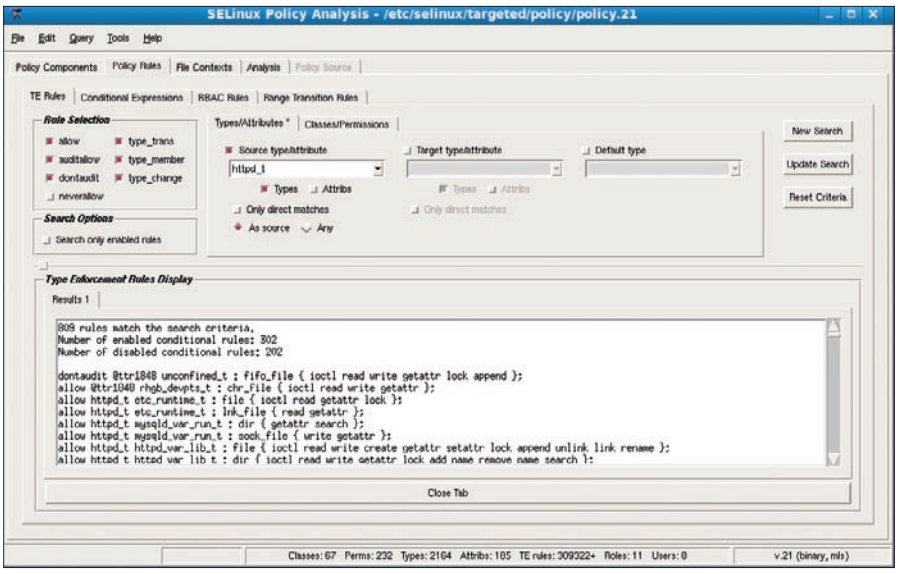

Figure 6: The graphical apol tool displays the binary policy in clear text.

Development

If you prefer to contribute actively to SELinux, rather than just configuring the SELinux policy, Fedora 8 has a number of tools that let you do so. For example, you can modify the binary policy on the fly with the *semanage* tool without accessing the sources. Of course, you can change every single property in this way, but this approach is typically best for simpler changes.

The SELinux policy gives the Apache web server the ability to bind to specific network ports. These ports are designated as *http_port_t* types in the SELinux policy. Calling *semanage* tells you which ports have a label of this kind:

Listing 4: sestatus -b

```
SELinux status:                 enabled
SELinuxfs mount:                /selinux
Current mode:                   permissive
Mode from config file:          permissive
Policy version:                 21
Policy from config file:        targeted

Policy booleans:
allow_console_login             off
allow_cvs_read_shadow           off
allow_daemons_dump_core         on
allow_daemons_use_tty           on
allow_execheap                  off
allow_execmem                   on
...
```

```
# semanage port -l |➥
 grep http_port_t
tcp 80, 443, 488, 8008, 8009
```

An administrator who wants to apply this label to a new port calls *semanage* as follows:

```
semanage port -a -t ➥
http_port_t -p tcp 777
```

Running *apol* against the policy to find a matching rule reveals the following:

```
allow httpd_t ➥
http_port_t : tcp_socket➥
 { name_bind name_connect };
```

The rule allows the processes in the *httpd_t* domain to access any network ports that have the *http_port_t* label;

Listing 5: semodule -l

```
amavis          1.3.1
amtu            1.1.0
apcupsd         1.1.2
audio_entropy   1.1.0
awstats         1.0.0
bitlbee         1.0.0
calamaris       1.2.0
ccs             1.2.0
cdrecord        1.2.1
certwatch       1.0
cipe            1.3.0
clamav          1.5.1
...
```

these ports are now 777, 80, 443, 488, 8008, and 8009.

If you would like to take this idea a step further and create completely new modules of your own, two tools are available to you: *system-config-selinux* and *policygentool*. *polciygentool* is part of the *selinux-policy-devel* RPM located in the */usr/share/selinux/devel* folder. It helps you create the files you will need to generate a binary policy module. A description of the individual steps could easily fill a book, so I'll just give you a rough overview of the required instructions in the following section.

Creating a Policy

First of all, the admin launches a tool and passes in the name of the application you want to protect, along with the name of the policy module to be created:

```
./policygentool foo ➥
/usr/bin/foo
```

The tool prompts you for a number of details about the application, such as whether it uses an init script, where the log files are stored, and so on.

After you have answered all of these questions, *policygentool* creates three files – *foo.fc*, *foo.if*, and *foo.te* – which are the file context, type enforcement, and interface files. The file context file allows the administrator to link the application files to an SELinux label; the type enforcement file specifies the matching rules – that is, what the application is allowed to do. The interface file places macros at the disposal of other policy modules.

After you have edited the files, you can then create the policy module with the following entry:

```
make -f /usr/share/selinux➥
/devel/Makefile
```

After this step, you should find the new *foo.pp* file below the current directory. Issuing the *semodule -i foo.pp* command loads the module into the kernel-based security server.

If this whole process sounds too complex for you, you can always use the graphical front end, *system-config-selinux*, to create a new policy module. An extensive tutorial for doing so is available online[4].

Listing 6: seinfo

```
Statistics for policy file: /etc/selinux/targeted/policy/policy.21
Policy Version & Type: v.21 (binary, mls)

   Classes:            67    Permissions:       232
   Sensitivities:       1    Categories:       1024
   Types:            2166    Attributes:        185
   Users:               8    Roles:              11
   Booleans:          115    Cond. Expr.:       144
   Allow:          171807    Neverallow:          0
   Auditallow:         28    Dontaudit:      134379
   Type_trans:       3286    Type_change:        88
   Type_member:        14    Role allow:         16
   Role_trans:          3    Range_trans:       158
   Constraints:        59    Validatetrans:       0
   Initial SIDs:       27    Fs_use:             17
   Genfscon:           66    Portcon:           292
   Netifcon:            0    Nodecon:             8
```

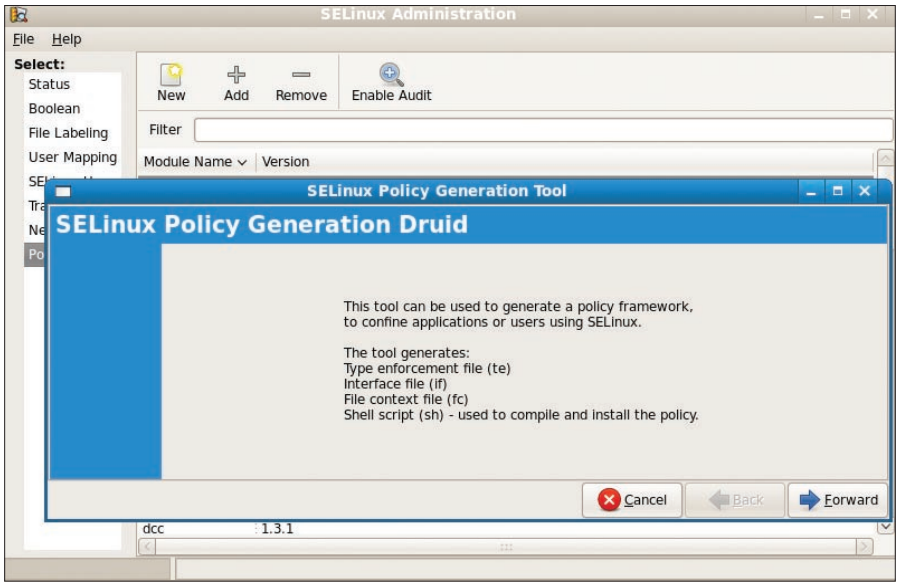

Figure 7: system-config-selinux makes it easy to create a new policy module.

Conclusions

SELinux is a vey useful security extension. Once it is activated, SELinux runs more or less transparently in the background, monitoring the running system – as long as the distributor has paved the way by providing a policy worthy of that title. As of this writing, Fedora is the leading distribution in this respect.

Recent releases have improved the usability of SELinux; for example, the SELinux logs are easier to read than before with the *setroubleshootd* tool. Even inexperienced users can develop their own policy modules to place new programs under the protective shield of SELinux, with a little help from the graphical front end, *system-config-selinux*. ■

INFO

[1] NSA SELinux website: *http://www.nsa.gov/selinux*

[2] Russell Coker's SELinux Debian play machines: *http://www.coker.com.au/selinux/play.html*

[3] Dan Walsh, Creating a Kiosk Account: *http://danwalsh.livejournal.com/13376*

[4] "A Step-By-Step Guide to Building a New Policy Module," by Dan Walsh, *Red Hat Magazine*, August 2007: *http://www.redhatmagazine.com/2007/08/21/a-step-by-step-guide-to-building-a-new-selinux-policy-module.html*